UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/663,455	09/15/2003	David Abu Ghazaleh	RSW920030055US1	2196

75532 7590 11/10/2009

LEE LAW, PLLC
IBM CUSTOMER NUMBER
P.O. BOX 189
PITTSBORO, NC 27312

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
11/10/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/663,455	GHAZALEH ET AL.	
	Examiner	Art Unit	
	TUAN A. VU	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *7/31/09*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,2,4-22 and 25-35* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,2,4-22 and 25-35* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 7/31/09.

As indicated in Applicant's response, claims 1, 2, 4, 6, 9, 16, 19-22, 25, 27-29, 31-35 have been amended. Claims 1-2, 4-22, 25-35 are pending in the office action.

Claim Rejections - 35 USC § 112

2. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

3. Claims 1-2, 4-22, 25-35 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

Claim 1 recites steps of providing a GUI, selecting an object, drawing a main object symbol, drawing a symbol of objects assigned to the main object, drawing a line between said objects. The claim as a whole amounts to providing a GUI tool serving as a interactive setting intended to respond or support actions taken or operations upon structures or objects from the what is recited as *providing*, and *responding* to user input step. The steps actions based on responding to the user inputs are recited as:

(3) *drawing, via the GUI, , using said specially programmed computer, a symbol corresponding to the main object of the program and labeling the symbol with a label descriptive of the object's features so that it is distinguishable from other symbols of a same object type;*

(4) *for each object assigned to or defined within the main object of the program, drawing, via the GUI, , using said specially programmed computer, a symbol corresponding to that object and labeling the symbol with a label descriptive of the object's features*;

(5) *drawing, via the GUI, , using said specially programmed computer, a line between each symbol drawn in step (4) and another symbol in the graphical representation corresponding to another object to which said object corresponding to said symbol drawn in step (4) is assigned or within which the another object is defined*, and

(6) *graphically denoting, via the GUI, the symbol in the diagram corresponding to the main object of the program so as to distinguish it from other symbols in the diagram by drawing an additional symbol around the symbol corresponding to the main object of the program*

However, the Specifications discloses a designer or a human architecture being provided with the GUI for exercising his/her intentions to drag or drop a mouse, for example drawing a connecting line, creating a symbol, adding more objects to relate to a main object, and to denote what is main or what is supplemental to a main object via GUI standards components for editing, or graphically drawing.

Indeed, the teachings gathered from Specifications are construed as steps performed manually; specifically by a human developer or architect (Specifications: *user can drag and drop* – 3rd para pg. 4; *preparer, architect ... by drawing* - pg. 5 bottom; pg. 7 bottom; middle and last para - pg. 8; bottom, pg. 13; *if desired ... may be drawn* – top para, pg. 15; *one may double click* – middle pg. 15; *mouse click, architect may decided to show* – middle pg. 16; bottom pg. 17; *mouse click* – middle pg. 19; *by the user* – bottom pg. 19; *double click* – middle pg. 20; *one may choose ... he or she would create* – 2nd para pg. 21; *user clicking* - bottom half pg. 21, top pg. 22

). None of the *drawing* instances (e.g. 'is drawn' 'may be drawn' 'are drawn' 'by first drawing' 'by drawing') found in the Specifications spell out or reasonably entail an automated act of drawing achieved by underlying code in execution (*using said specially programmed computer*) independent from user-driven events and/or in total absence of any human intervention. And this manual/human aspect of 'drawing' is all the more evident when the graphical interface is disclosed (Summary: 3rd para pg. 4) as presentation 'within which *the user can* drag and drop various symbols … and *interconnect them* and fill in the object attribute'. The inventor is deemed not in possession of any hardware and/or software means equipped with functions and structures in place that would clearly establish 100% functional/active realization of the steps of *labeling, denoting* and *drawing* in terms of automated processes precluding any trace of user control via intended use. One would not be able to make use of the invention when the keys actions taken and (functional elements: selecting, drawing) operating upon the provided GUI environment (or static elements) are manually achieved by a human being, which cannot be a part integral to the invention. The above steps (3) to (6) would be treated as though the interface thus provided operates as a container to receive results/events from user action within the interfacing environment, and provide obvious response thereto.

That is, steps (3) to (6) use the fabric of the GUI tool whereas the act of (3) to (6) are actually driven by the human's intervention which is not dependent of software sequential running. The inventor is in possession of software tool with GUI drawing means to enable the human to accomplish the acts recited in (3) to (6). No weight will be given to the precise acts recited from (3) to (6) because these steps emanate from the whims and desire of a human developers as no control either by the inventor's software can predict how or how many objects

are created, added or how (or how many) lines are joined or how symbol/objects are denoted, highlighted. That is, the steps (3) to (6) are not software instructions in sequential execution automated (via a computer runtime) in a typical way understood by one of ordinary skill in the art; but rather human driven actions based solely on human's will, intentions and personal capacity/skill/preference or intellectual faculties. Design intentions or actions performed by a human will not be part of the Invention, and steps (3) to (6) will be treated as obvious support from a GUI tool (emphasis added) when a user uses the tool to create graphical objects or joining lines among objects. Limitations (3) to (6) will be given weight as to graphical support enabling users to perform the users' design tasks; but no real merits will be given to the Inventor works in terms of executing those human acts as design choice or preferences cannot fall under the realm of software code product. The user's intention/preference/desire to draw, denote, to join lines will be construed as *intended use* when offered with the Inventor's proffered GUI tool; and the degree at which the user's actions vary in complexity and magnitude (number of symbols, of connecting lines, of objects created or denoting labels, main objects, secondary objects) would fall under *design choices*: none of intended use and design choice considered patentable subject matter.

The inventor is not given patentable merits for the actual steps recited in (3) to (6); that is, only the providing (1), the receiving (2) and the responding step (in response to users actions – drawing, labeling, distinguishing, connecting with lines, denoting a symbol) will be given merits.

Claims 4 and 9 amount to actions under the whim and control of a human and will be treated as a mere GUI support to enable the human choice to realize; i.e. no patentable weight given to design choice for the same reasons set forth against the language of claim 1.

Claims 2, 4-21 are rejected lack of proper disclosure or failing to remedy to the lack of enablement support set forth above.

Claim 22 recites s (third, fourth, fifth, sixth) executable instructions to 'cause the computing device, within the GUI' to:

(i) "to *draw* a symbol corresponding to the main object of the program and label the symbol with a label descriptive of the object's features so that it is distinguishable from other symbols of a same object type;

(ii) to detect drag and drop mouse actions … and *label* the symbols with a label descriptive of the corresponding object's features;

(iii) to *draw* a line between each symbol dropped in the workspace by the fourth computer executable instructions and another symbol in the workspace corresponding to another object to which said object corresponding to each symbol dropped in the workspace by the fourth computer executable instructions is assigned or within which the another object is defined; and

(iv) to graphically denote the main object of the program in the diagram *by drawing* an additional symbol around the symbol corresponding to the main object of the program."

As set forth above from studying the Disclosure to see whether the denoting/labeling and drawing are automated by a software product, it has been identified that labeling (ii), and drawing (i and iii) , denoting (iv) – via mouse clicking, dragging -- are mainly manual actions by a human architect or developer based on the Specifications, as this has been observed in the lack of enablement specific steps/actions of method claim 1 --refer to (3) to (6) from above. The actions recited as (i) to (iv), for analogous reasons, will not be given full patentable weight because they amount to actions initiated, taken, controlled by, and accomplished by the mind,

whim, desire of a user (all of which falling under the category of design choice and human intended use) as none of which is actually possessed by or pertinent to the inventor GUI or his work. The Disclosure presents no software instructions while executing, perform (in its independent runtime context) and achieve (independent of any user's whim) the various acts of (i) → (iv); rather software as disclosed is a mere GUI tool that, when used by a human, supports indirectly the human's completion of these acts, i.e. a conventional GUI tool concept. The extent to which the acts (i) to (iv) are given merits would be the structural and obvious support provided by as GUI tool recited as responding to and detecting events (drag/drop) based solely on a human action, i.e. the GUI for supporting the human's actions understood and expressed in (i) to (iv) being given weight as a structural settings where underlying software is made to respond (emphasis added) to an intended use. *Intended use* by the user when provided with the GUI tool and the *design choices* made by the user when developing objects, lines, symbols and graphical denotation (e.g. actual acts of *dragging, labeling, drawing, denoting*) thereof will not be given weight.

Claim 22 is rejected for failing to comply with the enablement requirement. Claims 25-35 fail to cure to the lack of enabling support set forth above, will be rejected for the same reasons.

Claim 25, 27, 28, 29, 32, 34, 35 recite 'cause the computer to", whereas the "cause to" acts of "enclose" (cl. 25), "prepare" (cl. 27), "include"(cl. 28), "specify" (cl. 29), , "provide" (cl. 34), "insert" (cl. 35) are not disclosed in the Specifications as software-driven automated actions provided directly by the Inventor, but rather as a human steps using the GUI; nor is there enablement for instructions that *restrict the computer device* ("restrict the computer device" - cl. 32) as in claim 32, when "operations" by an user is taught as being restricted by the tool. The

above claimed acts will be treated as *intended use* realized by the human developer based on his/her design choices (hence, with no real weight but rather as obvious acts), and the "restrict" limitation will be given broadest interpretation in view of well-known arts.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-2, 4-22, and 25-35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bailey (USPN: 6,701,513) in view of Kodosky (USPubN: 2003/0184580) further in view of Visio 2000 Standard Edition User Guide (Published by Visio International 1999 hereinafter Visio)

As per claim 1, Bailey discloses a method for graphically representing object oriented programming logic, the method comprising the steps of:

(1) providing, via a specially programmed computer, a graphical user interface (GUI) presenting a plurality of different symbols for use in a diagram of object oriented programming logic, each different symbol representing a different type of object in object oriented programming (e.g. 402, 402a fig. 4A; 430a, 440, 436a – Fig. 4D; *object members ... class methods ... object classes* - col. 7, lines 57-67; Col. 8, lines 24- 26);

(2) receiving a selection, via an input device of the specially programmed computer, of an object (e.g. Form 1 object 404 – Fig. 4D; col. 10 lines 36-40, 53-55) of the logic of a program to be represented in the diagram;

in response to the received selection within the GUI using said specially programmed computer automatically support drawing a symbol (e.g. object 404 Fig. 4D; e.g. col. 10, lines 53-55, 58-59, 64-65; col. 11, lines 35-38; 430a, 426a, 434a - Fig. 4D – Note: every icon reads on *object of the program*, since object are graphically instantiated to represent a software element – see col. 7 li. 24-39); labeling (e.g. col. 10, line 53 to col. 11 line 2; col. 11, lines 35-38; 426a 434a – Fig. 4D); connecting line (e.g. col. 4, lines 23-26; link 440 – Fig. 4D; 426, 434 -Fig. 8A; *graphically linking* - col. 13 lines 8-20; elements 902, 914, 026, 916. 918 - Fig. 9; col. 4 lines 29-50), denoting a symbol of one object to distinguish it from others (Form 426a, Fig. 4c; Form 434a, Fig. 4d).

Bailey does not specifically teach that said response to a selection comprises responses such as:

(3) drawing, via the GUI, , using said specially programmed computer, a symbol corresponding to the main object of the program and labeling the symbol with a label descriptive of the object's features so that it is distinguishable from other symbols of a same object type;

(4) for each object assigned to or defined within the main object of the program, drawing, via the GUI, , using said specially programmed computer, a symbol corresponding to that object and labeling the symbol with a label descriptive of the object's features;

(5) drawing, via the GUI, , using said specially programmed computer, a line between each symbol drawn in step (4) and another symbol in the graphical representation corresponding to another object to which said object corresponding to said symbol drawn in step (4) is assigned or within which the another object is defined, and

(6) graphically denoting, via the GUI, the symbol in the diagram corresponding to the main object of the program so as to distinguish it from other symbols in the diagram by drawing an additional symbol around the symbol corresponding to the main object of the program.

But the above steps (3), (4) (5) and denoting of objects to distinguish it over other objects as in (6) are construed as human acts, such as shown in Bailey (col. 4 lines 36-47; col. 5 lines 3-15; col. 8 li. 12-27; col. 9 li. 43 to col. 12 line 62); i.e. based solely of designer's choices, user's intention/preferences in the context of intended use and design choices, respectively. Based on the rationale set forth in the USC 112 Rejection, the above limitations will be given weight to the extent of the GUI support being obvious in light of the intended use such as via realization by human acts, when the intended use and design choices seem the driven factors for any realization of (graphical developing) results based on said GUI tool. The support of Bailey's tool to respond to the user's mouse or keyboard inputs or user events is strongly evident from Bailey (see Fig. 2, 4A-D, 6, 7-16); the support for realization of steps (3) to (6) in a broad sense is deemed disclosed, wherein the specific way by which the acts of (3) to (6) is to be realized would have been **obvious**. It would have been obvious for one skill in the art at the time the invention was made to implement the Bailey's graphical interface as set forth above, so that drawing main object with distinct label/symbol for it as in (3), drawing more objects as (4) assigned to a central object, adding lines to join these as in (5) and denoting a main object as in (6) to distinguish it with a label over other objects would all be supported by Bailey's tool based on the settings disclosed via interfaces components, toolbar, views, editor menu (see Fig. 4, 6) in light of the user's options to label, create, add lines, and populate icon legend as taught in Bailey from above, because by providing user's possibility to select a main object, add more objects to it,

connecting lines and graphically labeling the objects in a meaningful relationship, the development by Bailey would be more effective in realizing the intended paradigm where dependency among entities underlies needed functionality of the targeted software program for which Bailey's Gui tool is purported to do develop (see SUMMARY, col. 3-5)

Bailey does not explicitly teach selecting an object in (2) as a *main object* of the logic to be represented in the diagram and drawing a symbol corresponding to the main object and denoting it *so as to distinguish it from other symbols in the diagram by drawing an additional symbol around the symbol corresponding to the main object of the program* as in (6). Kodosky, in analogous graphical development endeavor, teaches creating an icon (e.g. paragraph [0011] lines 10- 12) representing a main program in a hierarchical view of the system (e.g. paragraph [0012] lines 9-11, 24-25). The support of a GUI for a user to denote one objects to make it more distinctive over other objects would have been obvious in Bailey. Based on Kodosky teaching of this, it would have been obvious to one of ordinary skill in the art at the time of the invention was made to combine the teachings of Bailey and Kodosky because Kodosky's teaching of main object with symbol would help user to easily understand logic that was described in detail according to specification of the system without creating any confusion. However, Kodosky combined with Bailey, does explicitly disclose drawing another symbol around the symbol corresponding to the main object

Visio teaches drawing another symbol around the symbol for the main object (pages 15-17). Based on a visual tool wherein highlighting of an object of interest or icon from which to draw more connected associations in Bailey (see Fig. 4A-D) and drawing a square-shape outline around an object for easy focus, it would have been obvious to one of ordinary skill in the art at

the time of the invention was made to combine the teachings of Kodosky (in light of Bailey) and Visio such that user or developer can easily create, distribute and/or deploy application with identifying various components in a distributed system as a main component in the system.

As per claim 2, Bailey (in view of the rationale in claim 1) discloses comprising the step of: providing, via the GUI, a plurality of additional different symbols for use in the diagram, each of the additional different symbols representing a different object oriented programming element type (e.g. col. 4, lines 23-26; *wire object class, Wire1, Wire2, wire control object* – col. 14 lines 21 to col. 15 line 49; elements 902, 914, 026, 916. 918 - Fig. 9) other than an object.)

As per claim 4, Visio teaches the method, wherein step (6) comprises (GUI support and response for) drawing a circle completely enclosing the symbol of the main object within the GUI (page 18, see shape-to-shape connections). The rationale of obviousness regarding enclosing a main object with a circle would be same as that set forth in claim 1.

As per claims 5-6, Bailey teaches the method, wherein the labels comprise text (e.g. col. 10, lines 53-55); wherein step (5) comprises drawing, via the GUI, the line between the symbol corresponding to the object defined in step (4) and the another symbol corresponding to the another object it is most directly assigned to or is most directly defined within (e.g. col. 4, lines 36-40).

As per claims 7 and 8, Kodosky teaches a method of visually creating a distributed system design and software programming which can be used in documenting software and to prepare a program specification (e.g, see *Print documentation* – Fig. 20A). One would be motivated to enable Bailey's framework building instance so that existing building instance or

pre-existed instances of build to be documented and distributed for further builds, data acquisition and testing (as suggested in Bailey: col. 3 lines 24-36)

As per claim 9, Bailey teaches (by virtue of the rationale in claim 1) the method, further comprising the step of: (8) repeating steps (1) - (5) to prepare a plurality of separate diagrams corresponding to separate parts of an overall application (e.g. col. 8, lines 14-18).

However, does not explicitly teach a first object is the main object appearing in at least a first one of the diagrams and is not a main object appearing in at least a second one of the diagrams.

Kodosky teaches a first object is the main object appearing in at least a first one of the diagrams and is not a main object appearing in at least a second one of the diagrams (e.g. paragraph [0015]). One would have been motivated to use Kodosky's main object to trigger the graphical enlistment of further objects in Bailey's endeavor, based on the connecting of icons as set forth in claim 1.

As per claim 10, Bailey does not disclose wherein the second diagram does not disclose objects assigned to and defined within the first object and the first diagram does disclose objects assigned to and defined within the first object. Based on the hierarchy of OO objects in Kodosky, and the well-known concept that base class include members defined therein in set forth in Baileys, the above submember of a base class would have flow out of the OOP approach in both Bailey (refer to claim 1) and Kodosky. Enabling a base class to disclose its defined member as opposed to the other way around (e.g. the second object not disclosing base/first object it is assigned to) would have been a obvious feature adopted in Bailey (in light of Kodosky)

As per claim 11, Bailey teaches the method, an application-level representation disclosing an overall software system (e.g. col. 8 lines 14-18)

As per claim 12, Bailey does not explicitly disclose wherein the label for the symbol corresponding to the first object in the second diagram identifies the first diagram as disclosing further details of the first object. Bailey teaches a GUI-based object-oriented enlistment of base objects (or graphical representation thereof) prior to expanding the base with submembers (or graphical representation thereof) wherein the label identifies as disclosing further details of the object (e.g. col. 9, lines 50-53). Enabling object in graphical representation of a base class to identifies itself even when such identification is shown in a base class or submember would have been obvious; and this is evidenced with OO nomenclature where a subclass (second diagram) has a label identifying a parent class, separated by a "." (label of a first object). Bailey teaches object-oriented building with labeling using this nomenclature (e.g. Label1.Caption, Form1.Label1 Fig. 8B), it would have been obvious for one skill in the art at the time the invention was made to implement the OOP by Bailey (in view of Kodosky) so that labeling would adopt this OO nomenclature such that the first class or object (or symbol representing it) is represented in the subclass or second object representation, i.e. the base object identifies itself in the labeling of the sub class.

As per claim 13, Bailey does not explicitly teach wherein the symbols representing different object types include at least the five following: a symbol for representing objects that are application type objects; a symbol for representing objects that are window type objects; a symbol for representing objects that are class type objects; a symbol for representing objects that are event script type objects; and a symbol for representing objects that are method type objects.

Bailey does teach plurality of objects or icons representing text boxes, radio buttons, scroll bars, menu bars and so on (e.g. col. 2, lines 3-9; col. 7, lines 57-61; col. 8, lines 8-11, 15-18, 24-37; Form.Label1, Form.Break1, Form.GreaterThan1, Form.CBAdd1, Form.TextBox - Fig. 14B-C; Form1.VSrollbar1 Form1.Label1 – Fig. 4D; col. 20 lines 32-67 – Note: icon being clicked for enabling user to specify/define event handler action OR Form1.xxx is analogized to even script type; or active icon symbols, such that a mouse event thereon reads on triggering code underlying such event); that is, each icon represents a corresponding object class that is available for use by developer (e.g. Fig. 9). It would have been obvious to one of ordinary skill in the art at the time of the invention was to add more graphical icons representing other OO programming types, i.e. symbols for application type, for window type, for class type, event script object, for method type as from above, which would improve readability of target application, container and contained elements relationship, event flows and membership of objects and corresponding meta-information or annotation specifying properties of members being developed, visibility of the OO hierarchical structuring of class members and their functional dependency with respect to a global scope application or another target subcomponent of a larger system type.

As per claim 14, Bailey does not explicitly disclose wherein the symbols representing different object types include at least the following five symbols: a first symbol for representing objects that are application type objects; a second symbol for representing objects that are window type objects; a third symbol for representing objects that are class type objects; a fourth symbol for representing objects that are event script type objects; and a fifth symbol for representing objects that are method type objects; and

wherein the symbols representing additional program elements include:

a sixth symbol for representing data transfer; a seventh symbol for representing databases; an eighth symbol for representing remote links; and a ninth symbol for representing inheritance.

However, these symbols are similar to those in claim 13 with added symbols; therefore, it is rejected for the same rationale as claim 13; and further Bailey teach (i) data transfer type and (ii) inheritance type symbols ((i) Form.Wait1, Form.User1; Form.Yield1 – Fig. 16; (ii) Form1.Variable1; Form1.CommandButton1 – Fig. 14A). Bailey disclose remote queries of COM objects (col 24 li. 48-65), hence it would have been obvious for one skill in the art at the time the invention was made to provide symbols denoting database type and remote link type because objects retrieved from a COM paradigm (Bailey: col. 8 lines 37-43) in light of the factory of objects (Bailey: col 7 lines 61-66) would necessitate links and database identifications.

As per claim 15, Bailey teaches the method, wherein the sixth, eighth, and ninth symbols are drawn connecting two other object symbols (e.g. col. 4, lines 35-40)

As per claim 16, Bailey does not explicitly disclose wherein the symbols representing different object types further include at least the following five additional symbols: a tenth symbol for representing objects that are menu type objects; a eleventh symbol for representing objects that are frame type objects; a twelfth symbol for representing objects that are button type objects; a thirteenth symbol for representing objects that are data structure type objects; and a fourteenth symbol for representing objects that are not one of the other object types. But based on the rationale of claim 13, and the way the elements are depicted in Bailey interface regarding general GUI tool icons that can be used by the user for data acquisition and manipulation of symbols (see icon 402, User interface, Data Acquisition tabs - Fig. 8B, 9) the

rationale as to render the provision of GUI icons to implement the above (tenth to fourteenth) symbols obvious for the same practical benefits required in a graphical development such as that of Bailey, based on the above and the benefits as set forth in claims 13 and 14.

As per claim 17, Bailey teaches the method, further comprising the step of: providing in a separate description of the logic to be performed responsive to an event script (e.g. col. 10, lines 9-12).

As per claim 18, Bailey teaches the method, wherein the symbol representing event script type objects is drawn connected to another object that directly executes the event script corresponding to the event script symbol (e.g. Form1.VSrollbar1 ←> Form1.Label1 – Fig. 4D).

As per claim 19, Bailey discloses interconnection of objects with line representing a relationship whereby interconnected objects do not invoke the other (see Fig. 9, 14A – Note: wire reads on not invoking) but does not explicitly teach wherein the fifth symbol representing method type objects is drawn connected to the main object of the program within the diagram and represents that the object is available within the main object of the program and does not represent that the main object of the program invokes it. The main class object linked to a secondary class has been taught in Kodosky, and the rationale as to joining OO objects with lines between base class and subclasses has been addressed in claim 1.

As per claim 20, Bailey discloses wherein step (1) comprises providing a graphical user interface in which a user is presented with a pallet containing the symbols (e.g. col. 7, lines 57-61). However, Bailey does not explicitly teach that receiving selection via the input device, in response to which steps (3) and (4) are performed, comprises: detecting dragging and dropping mouse actions associated with the symbols from the pallet into a work area. Analogous to

Bailey's mention of well-known tools s where drag and drop (e.g. Drag – col. 11 lines 36-44) are common practices (col. 2 lines 20-30) such as in Visual Basic COM technologies, Kodosky in an analogous endeavor, teaches a design tool that enables the user to drag and drop symbols from the pallet into a workspace (e.g. paragraph [0037]; [0034] lines 6-9). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention was made to combine the teachings of object-oriented manual manipulation in Bailey's tool using this well-known practice of drag-and-drop (evidenced in Kodosky) so that user or developer can easily visualize, select and manually join, add, and/or deploy iconic objects for implementing the logic or application based on the availability of various visual GUI components in a object-based visual developing tool, just as this drag-and-drop practice was practiced for its benefits by many visual developments technologies as set forth above, at the time the invention was made

As per claim 21, Bailey teaches wherein step (1) comprises providing a graphical user interface that receiving the selection via the input device, in response to which steps (3) and (4) are performed, comprises: detecting dragging and dropping the symbols from the pallet into a work area (e.g. col. 12, lines 35-37 – see above rationale), and wherein the labels comprise text (e.g. col. 10, lines 53-55) and further wherein at least some of the text labels can be made to appear in the graphical representation via an action taken by a user (e.g. col. 10, lines 63- 65).

As per claim 22, Bailey discloses a computer readable product embodied on computer readable media readable by a computing device for enabling a user to generate a graphical representation of object oriented programming logic (Abstract - lines 10-13; col. 3, lines 24-27; col. 7, lines 57-61; refer to claim 1), the product comprising first and second computer instructions to respectively:

(1) provide, via the computing device, a graphical user interface (GUI) in which a user is presented with a plurality of different symbols for use in developing a graphical representation of object oriented programming logic, each different symbol representing a different type of object in object oriented programming (e.g. col. 7, lines 24-39, 57-61; 402, 402a fig. 4A; 430a, 440, 436a – Fig. 4D; *object members ... class methods ... object classes* - col. 7, lines 57-67; Col. 8, lines 24- 26);

(2) cause the computing device to receive a selection of one and only one object in the diagram of the logic of a program represented in the diagram (refer to claim 1);

and in response to the received selection, via the computing device, within the GUI, instructions to automatically support drawing a symbol (e.g. object 404 Fig. 4D; e.g. col. 10, lines 53-55, 58-59, 64-65; col. 11, lines 35-38; 430a, 426a, 434a - Fig. 4D – Note: every icon reads on *object of the program*, since object are graphically instantiated to represent a software element – see col. 7 li. 24-39); labeling (e.g. col. 10, line 53 to col. 11 line 2; col. 11, lines 35-38; 426a 434a – Fig. 4D); connecting line (e.g. col. 4, lines 23-26; link 440 – Fig. 4D; 426, 434 -Fig. 8A; *graphically linking* - col. 13 lines 8-20; elements 902, 914, 026, 916. 918 - Fig. 9; col. 4 lines 29-50), denoting a symbol of one object to distinguish it from others (Form 426a, Fig. 4c; Form 434a, Fig. 4d).

Bailey does not explicitly disclose support in terms of third, fourth, fifth, sixth instructions to, respectively:

(3) cause the computing device to draw a symbol corresponding to the main object of the program and label the symbol with a label descriptive of the object's features so that it is distinguishable from other symbols of a same object type;

(4) cause the computing device to detect drag and drop mouse actions associated with symbols corresponding to object-oriented programming objects into a workspace and label the symbols with a label descriptive of the corresponding object's features;

(5) cause the computing device to draw a line between each symbol dropped in the workspace by the fourth computer executable instructions and another symbol in the workspace corresponding to another object to which said object corresponding to each symbol dropped in the workspace by the fourth computer executable instructions is assigned or within which the another object is defined; and

(6) cause the computing device to graphically denote the main object of the program in the diagram by drawing an additional symbol around the symbol corresponding to the main object of the program.

But the above responses (3), (4) (5) and denoting of objects to distinguish it over other objects as in (6) are construed as response to human acts, such as shown in Bailey (col. 4 lines 36-47; col. 5 lines 3-15; col. 8 li. 12-27; col. 9 li. 43 to col. 12 line 62); i.e. based solely of designer's choices, user's intention/preferences in the context of intended use and design choices, respectively. Based on the rationale set forth in the USC 112 Rejection, the above limitations will be given weight to the extent of the GUI supporting functionality being obvious in light of the intended use such as via realization by human acts, i.e. when the intended use and design choices seem the driven factors for any realization of (graphical developing) results based on said GUI tool. The obvious support as to how Bailey's (or any) GUI tool detects and responds from within the interface to human *design choices and intended use* has been set forth in claim 1, regarding the settings as how such computer-based GUI would react to user acts as

specific as (3), (4) (5) and denoting of objects to distinguish it over other objects as in (6); and such reactive support would have been obvious in light of the rationale therein.

Nor does Bailey explicitly teach selecting an object in the second instructions (2) as a *main object* of the logic to be represented in the diagram and drawing a symbol corresponding to the main object and *denoting* it *so as to distinguish it from other symbols in the diagram by drawing an additional symbol around the symbol corresponding to the main object of the program* as in the sixth instructions (6). This has been addressed as obvious based on the rationale using Kodosky (for denoting a main object) in light of Visio (for drawing an additional symbol around the main object) being set forth in claim 1.

As per claims 25-26, refer to claims 4-5 (Note: refer to the USC 112 Rejection for weight given to user's actions).

As per claim 27, Bailey discloses seventh computer readable instructions that cause the computing device to prepare a plurality of the diagrams corresponding to separate parts of an overall application and further comprising computer readable instructions that cause the computing device (Note: refer to the USC 112 Rejection for weight given to user's actions) to specify relationships between individual ones of the diagrams ((e.g. col. 8, lines 14-18; col. 9, lines 53-57; col. 16 line 5 to col 17 line 18; Fig. 14A-D).

As per claim 28, Bailey discloses wherein the seventh computer readable instructions comprise instructions which cause the computing device (Note: refer to the USC 112 Rejection for weight given to user's actions) to include references associated with symbols in one diagram identifying at least one other diagram within which the object represented by that symbol also appears (e.g Fig. 14A-D and related text).

As per claim 29, Bailey (with reference to claims 27-28) does not explicitly disclose instructions that cause the computing device (Note: refer to the USC 112 Rejection for weight given to user's actions) to specify in a first one of the diagrams the nature of the relationship of the representation of the object in the first diagram relative to the representation of the object in a second diagram, wherein the relationship between the object as represented in the first and second diagrams is selected from the group comprising: (1) the second diagram discloses additional details about the object in the first diagram; (2) the second diagram shows the object in a more abstract context than the first diagram; and (3) the object is the main object of the program of the second diagram.

But the object-oriented approach in Bailey (refer to claim 1) has been further evidenced with hierarchy of objects as in Kodosky whereby the OO relationships between the hierarchical object (as represented in the first and second diagrams) are selected from the group comprising of base objects and sub-objects. Based on the well-known concept of OO methodology, the limitations (1) where a subclass exposes some reference regarding its base class;(2) where the notation of a subclass shows global (non-detailed) reference of its base class nomenclature; and (3) where the base class is depicted in the diagram notation or representation of a subclass, would all have been obvious in view of the rationale of claims 9 and 10, wherein the diagram representing a derived or sub object not fully showing the internals of its parent object which it has been sub related to or derived from.

As per claim 30, Bailey does not explicitly teach wherein the symbols representing different object types include: a symbol for representing objects that are application type objects;

a symbol for representing objects that are window type objects; a symbol for representing objects that are class type objects; a symbol for representing

objects that are event script type objects; and a symbol for representing objects that are method type objects.

Bailey does teach plurality of objects or icons representing text boxes, radio buttons, scroll bars, menu bars and so on (e.g. col. 2, lines 3-9; col. 7, lines 57-61; col. 8, lines 8-11, 15-18, 24-37). Each icon represents a corresponding object class that is available for use by developer. It would have been obvious to one of ordinary skill in the art at the time of the invention was to add more graphical icons representing other programming objects which will improve the functionality of the system.

As per claim 31, Bailey does not explicitly teach the method, wherein the symbols representing different object types include: a first symbol for representing objects that are application type objects; a second symbol for representing objects that are window type objects; a third symbol for representing objects that are class type objects; a fourth symbol for representing objects that are event script type objects; and a fifth symbol for representing objects that are method type objects; and wherein the additional symbols representing additional program elements include: a sixth symbol for representing data transfers; a seventh symbol for representing databases; an eighth symbol for representing remote links; and a ninth symbol for representing inheritance.

Bailey does teach plurality of objects or icons representing text boxes, radio buttons, scroll bars, menu bars and so on (e.g. col. 2, lines 3-9; col. 7, lines 57-61; col. 8, lines 8-11, 15-18, 24-37). Each icon represents a corresponding object class that is available for use by

developer. It would have been obvious to one of ordinary skill in the art at the time of the invention was to add more graphical icons representing other programming objects which will improve the functionality of the system.

As per claim 32, Bailey teaches the method, seventh computer executable instructions that restrict the user to using the sixth, eighth, and ninth symbols to connect two other object symbols (e.g. col. 4, lines 35-40 – refer to the USC 112 Rejection).

As per claim 33, it is similar to claim 30 with added symbols; therefore, it is rejected for the same rationale as claim 30.

As per claim 34, Bailey discloses ninth computer executable instructions that cause the computing device to provide in a separate document a description of the logic to be performed responsive to an event script (e.g. col. 8, line 60 to col. 9 line 36; col. 20 lines 32-67 – Note: text enabling user to specify handler or properties using a mouse click on intended object read on separate description for response to event script).

As per claim 35, Bailey discloses:

tenth computer executable instructions that cause the computing device (Note: refer to the USC 112 Rejection for weight given to user's actions) to insert hidden text associated with symbols in the workspace made to appear in the workspace responsive to an action taken by a user (e.g. col. 10, lines 9-12, 53-55, 58-59, 64-65; col. 34, lines 16-20 – Note: object icons created on the Visual Basic, Visual J++, Cafe, ActiveX - see col. 7 lines 24-39 - disclose by virtue of inherency, underlying properties hidden for each icon and made available only when expressly invoked by user clicks).

Response to Arguments

6. Applicant's arguments filed 7/31/09 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

USC § 112 1st Rejection

(A) Applicants have submitted that the amended language has overcome the outstanding enablement rejection because response to user clicks and buttons by a computer program clearly evidences that the inventor is in possession of the claimed subject matter. The response to user's actions would have been obvious since any software-based GUI tool is purported as an interactive package made to respond to user taken initiatives. The Disclosure has no support for a non-user action that is executed solely by the inventor's software (i.e. to achieve the steps being rejected as non-enabled by the inventor), except for the obvious GUI response to any intended use of the users who utilizes the software to develop software modeling. The rejection will be maintained as set forth in the Office Action.

USC § 103 Rejection

(B) Applicants have submitted that prima facie by the Office action using Bailey/Kodosky has failed to disclose 'main object' being that of a program or main logic of a program (Appl. Rmrks pg. 19 top). Icon created in Bailey's GUI tool represent software functions or objects using a OO approach, and the logic of the software has to be modeled via the tool, hence each such icon represent object being part of the underlying software program being targeted by the development. The argument is not convincing and further purports to discuss on a newly claimed limitation, which is not entirely responsive to the previous Office Action.

(C) Applicants have submitted that claim 22 has 'drawing an additional symbol around … main object of the program' and claim 1 is adjusted to include the limitation (previously in claim

3) as to "denoting ... the symbol in the diagram ... so as to distinguish it from other symbols ... symbol around ... main object ..."; hence the Office action would be deficient in fulfilling these limitations (Appl. Rmrks pg. 19). The added language amounts to intended use and user's actions not taught as pertaining to the work of the inventor. The added subject matter has necessitated at least some modified grounds of rejection, for which Applicants have yet to overcome.

(D) Applicants have submitted that to fill the gap, the Office has attempted to use Visio, but Visio is not found as teaching 'drawing an additional symbol around ... symbol corresponding to the main object of the program' (Appl. Rmrks pg. 20) and that obviousness by the Office Action has proved to present no proper articulations or valid grounds for combining Bailey, Kodosky and Visio (Appl. Rmrks pg. 21). The creating of additional layer or symbol over a central iconic object has been taught in Visio. The act of adding a circle or to denote a key object with reinforced shading or color was a well-known concept as shown in Bailey, Kodosky, and Visio. The very act of adding a shade, a circle, or a thicker frame around an object belongs to the whim of the developer, and the software to support that intended use would have been obvious. No teaching provided by the Inventor is a stand-alone product executing free from intended use or design choice, especially when it comes to GUI action to draw object or to denote object. The argument is not sufficient to overcome the rejection, because support of Bailey tool to allow intended use (e.g. to denote a symbol with a thicker contour) would have been obvious, i.e. the USC 112 rationale notwithstanding.

(E) Applicants have submitted that prima facie regarding the obvious rejection against claims 2 ,5-21, 22, 3-4, 25 has not been properly established and that rejection against these claims

should be withdrawn (Appl. Rmrks pg. 22). No valid argument has been deemed persuasive to overcome the current state of Rejection, in view of the newly added language. The arguments amount to premature allegation for patentability when the grounds of rejection have been adjusted to address the newly submitted subject matter.

In all, the claims stand rejected as set forth in the above Action; and no patentable subject matter has been identified as a result of a non-enablement issue.

Interview Summary

7. Applicant's representative, att. Chris Lee, was approached via telephone contact (as per 11/03/09) to the effect that some clarifications (in regard to some claimed subject matter) were needed in such manner as to help the prosecution, because they would enable the Examiner to identify when a specific (or any potentially allowable) recited action has proper enablement by the Disclosure; e.g. to dispel the suspicion that the claimed functionality falls under well-known concepts or human usage, as opposed to being non-obvious teachings emanating from the inventor's product. But no agreeable terms was reached.

Conclusion

8. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37

CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

November 06, 2009